

REMARKS

Applicant respectfully requests the reconsideration of this application and the consideration of the following remarks.

Our records and the postcard returned from USPTO indicate that the IDS and the accompanying references, filed on 12/31/01, at least arrived at USPTO. For the convenience of the examiner, the references are submitted again in an IDS filed with the present response together with additional references that have not been previously submitted in this application. The exact publication date for the reference "Proposed SMPTE Standard for Television, SMPTE314M" is not shown on the IDS. According to the web site of SMPTE (e.g., http://www.smpte.org/smpte_store/standards/index.cfm?scope=1&CurrentPage=13&stdtype=smpte), SMPTE314M was issued in 1999. Thus, applicant believes that the reference "Proposed SMPTE Standard for Television, SMPTE314M" was published in 1999 or earlier.

Claims 13, 15, 17, 19, 22, 42, 61-62, 67-68, 74-75 and 77 were objected to because of informalities. The claim amendment in the present response removes the informalities.

Claims 14, 16, 18, 23, 26, 27 and 33 were provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1, 4, 5 and 11 of the copending Application No. 10/038,478. Applicant respectfully disagrees.

Each of claims 14, 16, 18, 23, 26, 27 and 33 specifically recites a particular arrangement of *an execution unit* in a microprocessor. Such arrangements for *an execution unit* in a microprocessor are not obvious in view of the methods recited in claims 1, 4, 5 and 11 of the copending Application No. 10/038,478.

Claims 12, 20-22, 48-52, 56, 70-71 and 76-79 were rejected for being indefinite. The amendment in the present response overcomes the rejection.

Claims 1-79 were rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Patent No. 6,397,324 (hereinafter "Barry"). Applicant respectfully disagrees.

Applicant respectfully submits that the pending claims are patentable over Barry.

For example, claim 1 recites:

1. (original) An execution unit in a microprocessor, the execution unit comprising:
look-up memory; and
a first circuit coupled to the look-up memory,
the first circuit, in response to the microprocessor receiving a first instruction, partitioning the look-up memory into a first plurality of look-up tables,
the first circuit, in response to the microprocessor receiving a second instruction, partitioning the look-up memory into a second plurality of look-up tables which are different from the first plurality of look-up tables.

The Office Action relied upon the L2TBL and L4TBL instructions of Barry and the description of Col. 7, lines 54-62, Col. 9, lines 63-67, Col. 11, lines 10-13, and Col. 11, lines 33-36 in Barry for the rejection of claim 1.

However, Barry does not show "*the first circuit*" partitioning the look-up memory into different look-up table configurations. Barry shows the designer's choices of split memories into separate banks which are addressable independently. For example, Col. 11, lines 33-36 of Barry shows:

"To support this instruction type, the SP and each PE data memories are split into four separate banks which are addressable independently." (Col. 11, lines 33-36 of Barry)

Clearly, this description is about a design choice of using four separate memory banks, which are addressable independent, in order to have SP and each PE data memories that support this instruction type (L4TBL). Such a choice of a designer at the design phase cannot be considered as corresponding to the functionality of “the first circuit”. A designer is not a circuit.

Furthermore, the designer’s choice to “split” the memory into separate memory banks at the design phase is not in response to the microprocessor receiving an instruction.

Thus, Applicant respectfully submits that Barry does not have “a first circuit” as specified in claim 1.

Further, for example, claim 2 recites:

2. (original) An execution unit as in claim 1 wherein a total number of bits used by each entry in the first plurality of look-up tables is different from a total number of bits used by each entry in the second plurality of look-up tables; and wherein the microprocessor is a media processor formed in a monolithic semiconductor substrate, which comprises a memory controller for controlling host memory, said media processor being coupled to said memory controller.

The Office Action asserted that “a L2TBL and a L4TBL instruction can use different number of bits for each entry.” However, such an assertion, even if correct, is about instructions. A person skilled in the art understands that instructions are not circuits. Barry does not show “a first circuit” which partitions memory into look-up tables of different entry sizes, in response to different instruction. Thus, Barry does not have an execution unit as claimed.

Further, memory banks 431/433 of Figure 4 of Barry are not host memory; and memory interface unit 485 of Figure 4 of Barry is not a memory controller for controlling host memory.

Further, for example, claim 3, recites:

3. (original) An execution unit as in claim 1 wherein a total number of entries in each of the first plurality of look-up tables is different from a total number of entries in each of the second plurality of look-up tables.

Col. 11, lines 13-14, of Barry shows “Maximum architecture defined size of the LUT is 64 K entries.” Col. 11, lines 44, of Barry shows “Maximum size of the LUT for this case is 256 entries.” Clearly, these are architectural limitations on the possible sizes of LUTs that can be implemented on the processor of Barry for different types of instructions. Such architectural limitations do not correspond to the functionality of “the first circuit”.

Further, for example, claim 4, recites:

4. (currently amended) An execution unit as in claim 1 wherein the look-up memory comprises a plurality of look-up units, and wherein the first circuit is to configure the plurality of look-up units into a third plurality of look-up tables in response to the microprocessor receiving a third instruction.

Barry does not show “*the first circuit*” configuring the plurality of look-up units into a third plurality of look-up tables. The Office Action argued that “The LTBL instruction partitions the look-up memory into a single look-up table”.

Clearly, “LTBL instruction” is not a circuit. Even if the assertion of “The LTBL instruction partitions the look-up memory into a single look-up table” were correct, it does not properly correspond to the claim limitation.

Further, the LTBL instruction does not cause the partitioning of the look-up memory into a single look-up table. The rejection relied upon Col. 10, lines 24-61, of Barry, which

describes the encoding of the LTBL instruction. This portion of the description of Barry contains no sufficient details regarding how the LTBL instruction is implemented. From Figure 4 of Barry and its associated description, which shows the *apparatus* according to Barry, one understood that Barry splits the memory into memory banks that are addressable independently *in the design* of the apparatus. The instruction LTBL does not “partition” the look up memory.

For example, claim 20 recites:

20. (currently amended) An execution unit in a microprocessor comprising:
- a plurality of look-up tables;
 - a first circuit coupled to the plurality of look-up tables, the first circuit configured to receive a string of bits;
 - a second circuit coupled to the plurality of look-up tables and the first circuit, the second circuit configured to receive a plurality of data elements, in response to the microprocessor receiving a single instruction, the second circuit generating a plurality of indices using the plurality of data elements and the string of bits, the plurality of look-up tables looking up simultaneously a plurality of entries using the plurality of indices; and
 - a third circuit coupled to the plurality of look-up tables, the third circuit combining the plurality of entries into a first result.

The Office Action relied on the description of the L2TBL instruction for the rejection of claim 20. However, claim 20 recites the particular arrangement of an execution unit in a microprocessor. The description on the input/output relation of the L2TBL instruction shows only the structural relations outside execution units.

Further, Barry does not show any execution unit receiving data from R_z register as a string of bits. Furthermore, in Barry, R_z register is partitioned into R_{ze} and R_{zo} by design, not by the L2TBL instruction. For the L2TBL instruction, the numbers stored in R_{ze}.H0 and R_{zo}.H0 are used to compute the addresses. There is no indication of receiving data from R_z register as a string of bits.

Similarly, Barry does not show *an execution unit* combines the plurality of entries into a result. The L2TBL instruction storing the results in an entry in the register file is not sufficient for the anticipation of a particular arrangement of an execution unit.

Further, claim 21 recites:

21. (original) An execution unit as in claim 20 further comprising:
a fifth circuit coupled to the second circuit, the fifth circuit configured to receive at least one format; and
a sixth circuit coupled to the fifth circuit and the third circuit, in response to the microprocessor receiving the single instruction, the fifth circuit formatting the string of bits into at least one escape data using the at least one format, and
the sixth circuit combining the at least one escape data with the first result into a second result.

Barry does not shown an execution unit that has a “fifth circuit” to format the string of bits into at least one escape data using the at least one format and a “sixth circuit” to combine the at least one escape data with the first result into a second result.

Barry indicates that the sign extension bit (S_x) in the instruction can cause the byte or half-word operands to be sign-extended to 32-bits (Col. 10, lines 39-41, Barry). However, if the sign-extended look up results of Barry were considered as corresponding to “escape data”, it is clear Barry does not show the combination of the sign-extended look up results with the results in register R_t (which were considered as corresponding to the first result) into

a second result. Further, since the Office Action considered “the string of bits” as corresponding to the indices in R_z, it is clear that the sign-extended look up results of Barry are not formatted from “the string of bits”.

Alternatively, if the sign-extended indices were considered as corresponding to “escape data”, it is clear that the sign-extended indices of Barry are not combined with the look up results in register R_t into a second result. The Office Action considered the results in the register R_t as the “first result”. There is no “second result” and no combination of the “escape data” with the “first result.”

Thus, the features of Barry could not be properly applied to the claim limitations for the rejection of claim 21.

The above discussion for claims 20 and 21 also applies to claims 47 and 76, claims 9-12.

Further, for example, claim 78 recites:

78. (original) An execution unit as in claim 76 wherein the at least one format is received from *an entry of a register file*.

The Office Action asserted that “the format is specifies in the “size” field of the instruction, and then the actual format (data of a certain data type) is read out from an entry in the register file.” Applicant respectfully submits that such an assertion is erroneous. Barry does not show *an entry of the register file* that specifies “the actual format.” It is improper to consider data of a certain data type, which is the data, as the format of the data. In the instructions of Barry, the sign extension bit (S_x) is used to indicate the format of sign-extension to 32-bits. It is clear the indication of sign-extension is received from the instruction of Barry. No format is received from an entry of a register file in Barry.

Further, for example, claim 79 recites:

79. (original) An execution unit as in claim 78 wherein the single instruction specifies an index of the entry in the register file.

It is clear that the sign extension bit (Sx) does not specify any particular register from which the format is received. The sign extension bit specifies the information need for the instruction of Barry regarding the format. No further information form an entry in the register file is need for the instruction of Barry.

Further, for example, claim 48 recites:

48. (currently amended) An execution unit as in claim 47 further comprising:
means for receiving a plurality of data elements specifying the plurality of segments in the string of bits.

The Office Action did not point what in Barry corresponds to “a plurality of data elements specifying the plurality of segments”. Even if the indices in Rze and Rzo were considered as corresponding to the string of bits, Barry’s instructions clearly do not use a plurality of data elements to specify the plurality of segments. The predetermined data fields in the Rze and Rzo registers are used for the instruction. Thus, there would be no “means for receiving a plurality of data elements specifying the plurality of segments ...” in any execution unit of Barry.

Further, for example, claim 49 recites:

49. (original) An execution unit as in claim 48 wherein the plurality of data elements are received from an entry in a register file.

The Office Action points to the data in Rze and Rzo for the rejection of claim 49. Applicant respectfully submits that it is improper to consider the data in Rze and Rzo as both the “string

of bits” and “the plurality of data elements”. Note that claim 48 recites “*a plurality of data elements specifying the plurality of segments* in the string of bits”.

Further, for example, claims 51-52 recites:

51. (original) An execution unit as in claim 48 further comprising:
means for receiving a bit pointer, wherein the plurality of segments in
the string of bits are determined using the bit pointer and the
plurality of data elements.

52. (original) An execution unit as in claim 51 further comprising:
means for generating a new bit pointer using the first result.

The Office Action considered the L2TBL instruction as a bit pointer in rejecting claim 51 and considered the look up result of a L2TBL instruction as a new bit pointer for subsequent instruction. Applicant respectfully disagrees. A person skilled in the art understands the term “pointer”. Since the L2TBL instruction does not encode any information about a pointer specified in the claim, the L2TBL instruction cannot be considered as corresponding to the bit pointer recited in the claim.

Furthermore, Barry does not show that the look up results in Rze and Rzo are used in the execution unit to further generate a new bit pointer. It is illogical to consider the look up results both as the first result and as the new bit pointer generated from the first result.

Further, claim 63 recites:

63. (original) An execution unit as in claim 47 wherein the means for
combining the plurality of entries comprises:
means for selecting a valid data from the plurality of entries.

In Barry, “double-word” data is not supported by the L2TBL instruction according to the architectural design; and no “double-word” data is looked up from the tables. The lack of the capability to support “double-word” is not an indication of selecting data that is not “double-word” from mixed data. There is no process of “selecting” in Barry. Thus, the argument in the Office Action is not based on the description in Barry. Nothing in Barry corresponds to “means for selecting ...”.

Further, claim 72 recites:

72. (currently amended) An execution unit as in claim 47 further comprising:
means for receiving a first number indicating a position of a last bit of
input in the string of bits.

In rejecting claim 72, the Office Action mistakenly mixed the description of LATBL instruction (Col. 12, lines 66-67) with the Figure for L2TBL instruction (Figure 6A) as the description for L2TBL instruction. Note that the LATBL instruction, Load Address of Table Item instruction, is very different from the L2TBL instruction, Dual Address Table Look-up instruction. In Barry, the format of the indices in Rze and Rzo, which were considered as the string of bits, is fixed for the L2TBL instruction. In Col. 10, lines 65-67, Barry clearly show that Rze.H0 and Rzo.H0 are used. The size field 23-22 does not indicate the last bit of the input string of bits.

Further, claim 73 recites:

73. (original) An execution unit as in claim 72 further comprising:
means for generating an indicator indicating whether any bit after the
last bit of input is used in obtaining the first result.

The Office Action asserted that “there is inherently an indicator which indicates if the data needs sign-extension, which would indicate that bits after the last bit (i.e., the sign-extension bits) have been used in creating the first result.” Applicant respectfully disagrees.

Applicant respectfully points out that “the last bit of input” corresponds to “a last bit of input in the string of bits” recited in claim 72. In Barry, the predetermined portions Rze.H0 and Rzo.H0 are used. There is no reason for Barry to use any indicator to indicate whether or not any bit in Rz is used in obtaining the look up results stored in Rt, if the limitations are considered in a consistent way.

Further, claims 74-75 recite:

- 74. (currently amended) An execution unit as in claim 47 further comprising:
means for generating an indicator indicating whether one of the plurality of segments of bits contains a predetermined code.
- 75. (original) An execution unit as in claim 74 wherein the predetermined code represents an end of block condition.

The Office Action pointed to Col. 15, lines 13-17 and Col. 15, line 63 – Col. 16, line 4 of Barry for the rejection of claims 74-75. However, this description of Barry is substantially irrelevant to claims 74-75, since claims 74-75 recite “the plurality of segments of bits”, which corresponding to “a plurality of segments of bits in the string of bits” recited in claim 47. Consider that the indices in Rze and Rzo were considered as corresponding to “the plurality of segments of bits”, it is clear that Barry does not have, in an execution unit, “means for generating an indicator indicating whether one of the plurality of segments of bits contains a predetermined code”, such as a code for an end of block condition.

Claims 16, 18, 26 and 27 recite:

16. (original) An execution unit in a microprocessor, the execution unit comprising:
a plurality of look-up tables;
a first circuit coupled to the plurality of look-up tables and a Direct Memory Access (DMA) controller, the first circuit, in response to the microprocessor receiving a single instruction, replacing at least one entry in at least one of the plurality of look-up tables with at least one data element using the DMA controller.
18. (original) An execution unit in a microprocessor, the execution unit comprising:
a plurality of look-up tables;
a first circuit coupled to the plurality of look-up tables and a Direct Memory Access (DMA) controller, the first circuit, in response to the microprocessor receiving a single instruction, replacing at least one entry for each of the plurality of look-up tables with a plurality of data elements using the DMA controller.
26. (original) An execution unit in a microprocessor, the execution unit comprising:
means for replacing at least one entry in at least one of a plurality of look-up units in a microprocessor unit with at least one number using a Direct Memory Access (DMA) controller;
wherein the above means operate in response to the microprocessor receiving a single instruction.
27. (original) An execution unit in a microprocessor, the execution unit comprising:

means for replacing at least one entry for each of a plurality of look-up units in a microprocessor with a plurality of numbers using a Direct Memory Access (DMA) controller;
wherein the above means operate in response to the microprocessor receiving a single instruction.

The Office Action relied upon the S2TBL instruction and the memory interface unit (485 of Fig. 4) of Barry for the rejection of claims 16, 18, 26 and 27.

The Office Action asserted that "... the memory interface unit (485 of Fig. 4), which is a DMA controller (see, Col. 7, lines 50-62)." Applicant respectfully submits that such an assertion is incorrect. A person skilled in the art understands that the memory interface unit 485 of Fig. 4 is not a DMA controller. There is no indication in Barry that the memory interface unit 485 of Fig. 4 is a DMA controller. In contrary, Fig. 1 of Barry shows a DMA controller 183 which is separate from the data memory interface 125.

Col. 7, lines 50-62, of Barry shows "the local data memories for the SP and each PE are currently organized as two memory banks to support independent, simultaneous accesses by the processing unit, and the direct memory access (DMA) controller". However, there is no indication in Barry about the type of access that involves a DMA controller.

Furthermore, there is no apparent relation between the DMA controller and the S2TBL instruction. Applicant respectfully submits that it is incorrect to assert that "the S2TBL instruction further specifies two pieces of data denoted by even and odd addresses (each pieces of data is considered a number) stored in the register file at Rte and Rto that will be written into the look-up table entries pointed to by the pointers created ...".

What is stored in Rte and Rto of Barry are the two pieces of data, not the addresses of the two pieces of data. The two pieces of data stored in Rte and Rto are copied into the memory banks in response to the S2TBL instruction. See, for example, the "operation"

column of the table in Fig. 8B. Thus, the S2TBL instruction loads data from the computer register file (CRF) into the memory banks (431 and 433), which would not involve a DMA controller; and the assertion in the Office Action is incorrect.

For example, claim 33 recites:

33. (original) An execution unit in a microprocessor, the execution unit comprising:
means for receiving a plurality of numbers;
means for partitioning look-up memory into a plurality of look-up tables;
means for looking up simultaneously a plurality of elements from the plurality of look-up tables, each of the plurality of elements being in one of the plurality of look-up tables and being pointed to by one of the plurality of numbers;
wherein the above means operate in response to the microprocessor receiving a single instruction.

The Office Action relied on Barry (Col. 7, lines 54-62 and Col. 9, lines 63-67) for the limitation of “partitioning look-up memory into a plurality of look-up tables”. However, the description of Col. 7, lines 54-62 and Col. 9 lines 63-67 of Barry does not correspond to the claim limitation, since claim 11 recites the limitation “the above operations are performed in response to the microprocessor receiving the single instruction”, which applies to “partitioning look-up memory into a plurality of look-up tables”.

The Office Action relied on the L2TBL instruction of Barry for the rejection of claim 11. However, the description of Col. 7, lines 54-62 and Col. 9, lines 63-67 of Barry does not show “partitioning” in response to the L2TBL instruction. Col. 7, lines 54-62, of Barry shows the hardware design choice of using “a multiple bank memory that makes it possible to generate multiple independent data-dependent load and store operations”. Col. 9, lines 63-

67, of Barry shows the design choices of using “both memory bank-1 431 and memory bank-0 433 simultaneously to support two load operations in parallel or two store operations in parallel”. These design choices of using a memory with multiple banks cannot be considered corresponding to “*partitioning* look-up memory into a plurality of look-up tables, ... wherein the above operations are performed *in response to the microprocessor receiving the single instruction*”.

Furthermore, a designer is not an execution unit. The designer cannot be considered as corresponding to “means for partitioning ...” in an execution unit of a microprocessor.

Note that, in response to the L2TBL instruction, the processor of Barry uses the base addresses stored in the address register file (ARF) and offsets stored in the computer register file (CRF) to access the multiple bank memory. Since the look-up tables in the multiple bank memory of Barry are imaginary, based on how one interprets the meaning of the base addresses and offsets, the processor of Barry does not perform “partitioning look-up memory into a plurality of look-up tables, ... wherein the above operations are performed in response to the microprocessor receiving the single instruction”.

Claim 14 and 23 recite the limitation of “wherein the microprocessor is a media processor integrated with *a memory controller for host memory* on a single integrated circuit”. The memory interface unit (485) of a Barry is not a memory controller for host memory. The memory interface unit (485) is for local memory banks (431 and 433). Barry does not show “a media processor integrated with a memory controller for host memory on a single integrated circuit”.

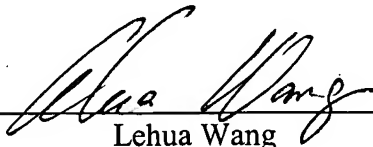
Other claims recite the limitations discussed above or indirectly contain the above discussed limitations through dependency to the above discussed limitations. Thus, the pending claims are patentable over Barry.

Please charge any shortages or credit any overages to Deposit Account No. 02-2666.
Furthermore, if an extension is required, Applicant hereby requests such extension.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Dated: 12/9, 2004

A handwritten signature in dark ink, appearing to read "Lehua Wang", is written over a horizontal line.

Lehua Wang
Reg. No. 48,023

12400 Wilshire Boulevard
Seventh Floor
Los Angeles, California 90025-1026
(408) 720-8300